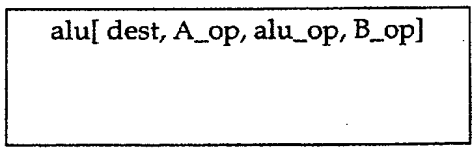

(ALU instruction)

Fig. 1A

A_Op and B_Op are symbolic registers, the following are possible register class assignment pairs to A_Op and B_Op:

A_Op	B_Op
General Purpose Register (GPR) (A Bank)	GPR (B Bank)
GPR (B Bank)	GPR (A Bank)
Transfer In (SRAM (S) or DRAM (D))	GPR (A Bank or B Bank)
GPR (A Bank or B Bank)	Transfer In (S) or (D)

(Possible register class assignment to A_Op and B_Op)

Fig. 1B

Dest ← Src_1 Op Src_2

(Intermediate Representation of Target Machine Instruction)

Fig. 2

$$M = \left\{ \left(s_i, c_{s_i} \right) \middle| \; 1 \le i \le N_s ; c_{s_i} \textit{ equals either } C_k (1 \le k \le m) \textit{ or } C \right\}$$

Definition of Register Class Assignment Map

Fig. 3

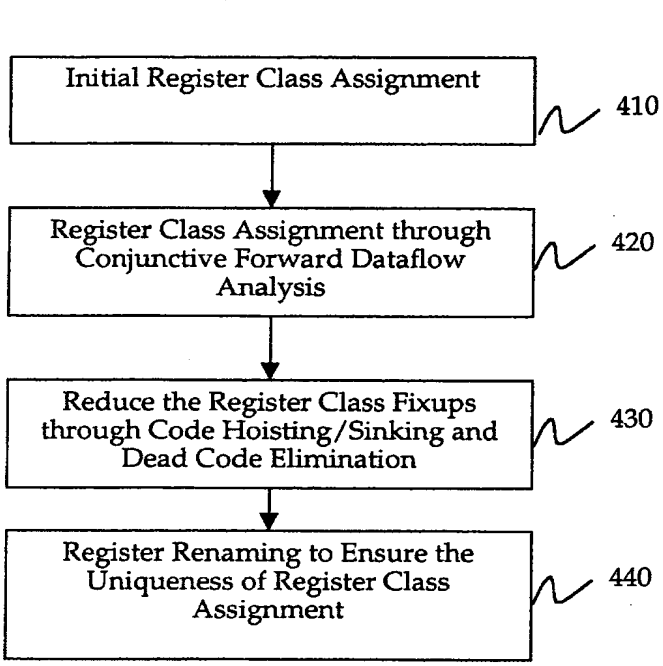
400
Initial Register Class Assignment
410
Register Class Assignment through Conjunctive Forward Dataflow Analysis
420
Reduce the Register Class Fixups through Code Hoisting/Sinking and Dead Code Elimination
430
Register Renaming to Ensure the Uniqueness of Register Class Assignment
440

Fig. 4

For each basic block $\mathbf{B_j}$

For each instruction **i** inside $\mathbf{B_j}$ from block entry to block exit

For each operant **O** of **i**

If **O** is a symbolic register $\mathbf{s_k}$

If $\mathbf{s_k}$ requires specific register class assignment in **i**

Regclass ($\mathbf{s_k}$,**i**) = $\mathbf{C_n}$, where $\mathbf{C_n}$ is the specific register class required by **i**;

Else

Regclass ($\mathbf{s_k}$,**i**) = **C** ;

Fig. 5

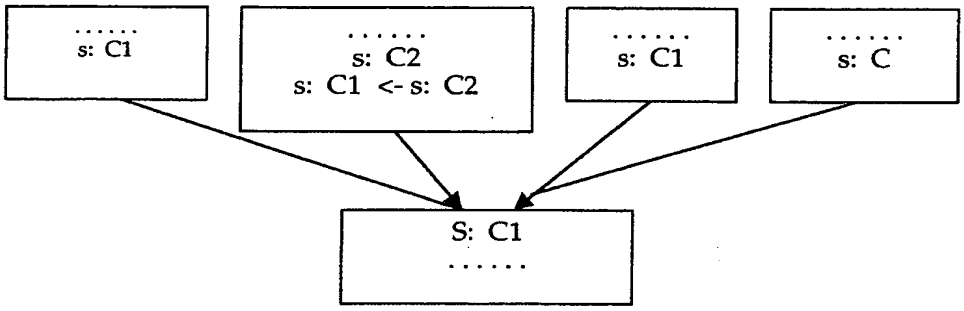

Fig. 6

```
OUT_M(i)=IN_M(i);
for each symbolic register operand s_k of instruction i (Suppose s_k is the Nth operand)
 Find out the value of PrevAssign where (s_k PrevAssign) ∈ OUT_M(i);
 CurAssign = Regclass (s_k, i);
 if (CurAssign = =C)
       if (PrevAssign! =C)
          if (IsValidRegClassAssignment (i, Nth, PrevAssign))
              Regclass(s_k i)=PrevAssign;
              continue;       /*continue the next loop iteration */
          else
              CurAssign=GetNextRegClass(Inst, NthOperand);
              If (s_k is not the destination operand)
                  Insert before i the register class fixup from PrevAssign to CurAssign;
       else
          CurAssign =GetNextRegClass(Inst, NthOperand);
       Regclass (s_k, i) =CurAssign;
       Replace (s_k PrevAssign) with (s_k CurAssign) in OUT_M(I) = = =>OUT_M(i);
 else
       if ((s_k, CurAssign) ∉ OUT_M(i))
              if (PrevAssign!=C AND s_k is not the destination operand)
                  insert before i the register class fixup from PrevAssign to CurAssign;
          Replace (s_k PrevAssign) with (s_k CurAssign) in OUT_M(i);
```

Fig. 7

For each basic block b based on the topological order of the dataflow graph

Calculate IN_M(b);

for each instruction i inside basic block b from entry to exit

Calculate IN_M(i);

Calculate OUT_M(i));

Calculate OUT_M(b);

Fig. 8

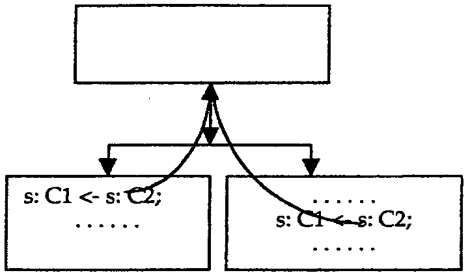

Fig. 9A (Hoisting)

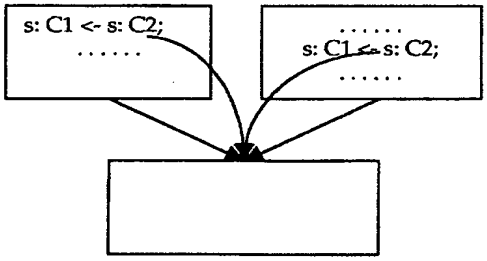

Fig. 9B (Sinking)

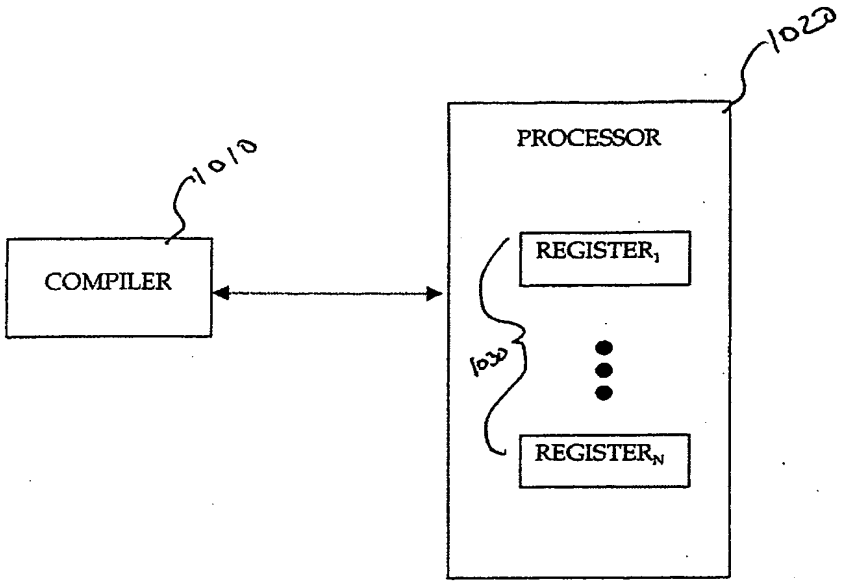

FIG. 10

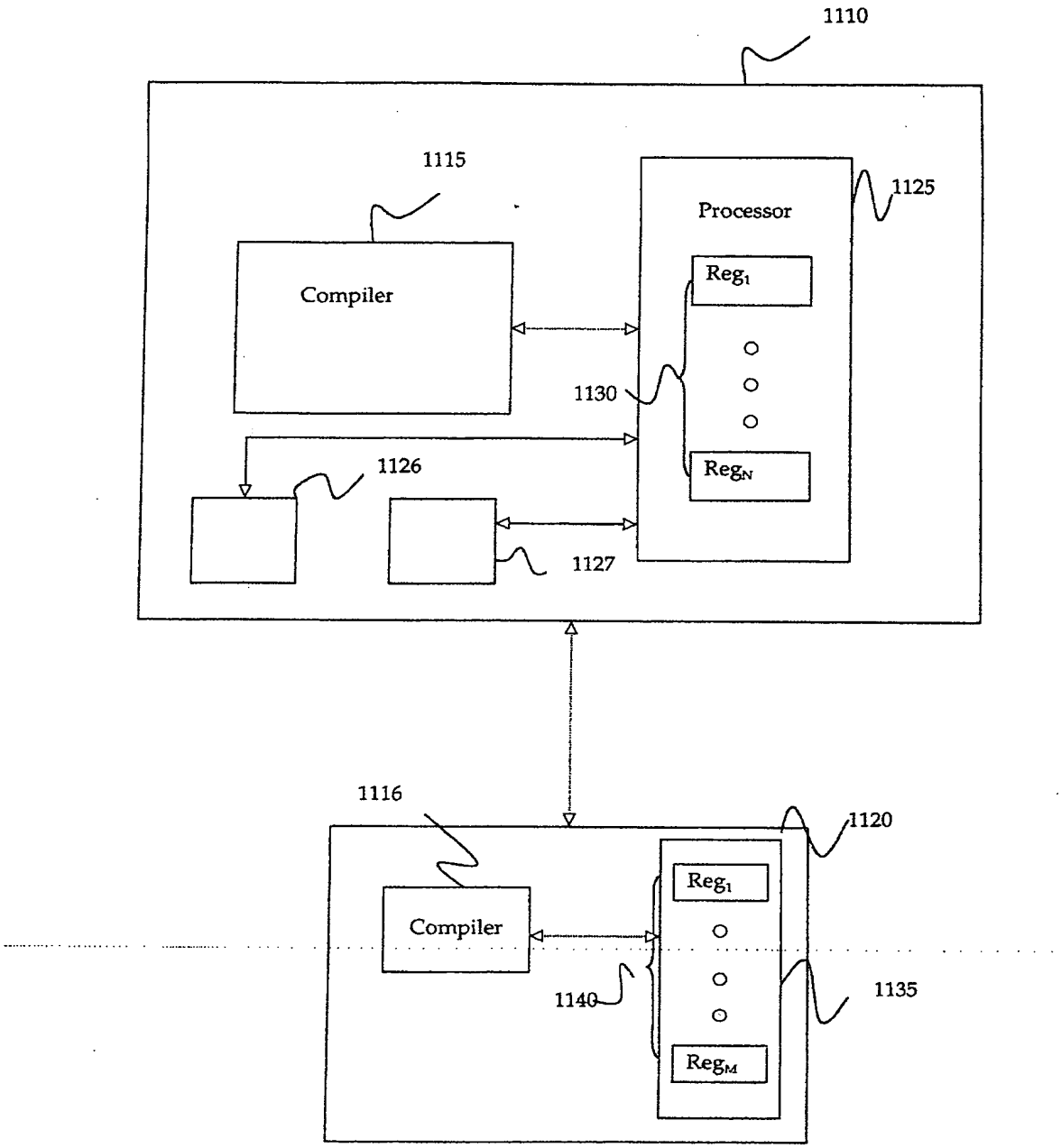
1110
1115
Compiler
Processor
1125
Reg1
1130
RegN
1126
1127
1116
Compiler
1120
Reg1
1140
1135
RegM

Fig. 11